Exercises

I.1 Consider the third solution for the mutual exclusion problem (take turns). Does the addition of the assignment *t := Y* (resp. *t := X*) after crossing the bridge solve the problem? Discuss this.

I.2 Which variables are shared/private in this program fragment?

$i := 0;$
while $i \neq 100$ do
    $x := x+1;$
    $i := i+1$
od
	$\|\|$	$j := 0;$
		while $j \neq 100$ do
		    $x := x+1;$
		    $j := j+1$
		od

Exercises

I.3 Consider the program in I.2. Given that the initial value of *x* is 0, what will the final value be? Assume that the assignment *x:=x+1* is atomic. (See also exercise I.5)

I.4 Consider the parallel execution of statements *P* and *Q*. Initially, variable *x* equals 0. What are the possible final values of *x* with

a. *P: x := 1* and *Q: x := 2*

b. *P: x := x+1* and *Q: x := x+2*

c. *P: y := x; x := y+1* and *Q: x := x+1*

Provide the traces. Assume atomic assignments.

Exercises

I.5 In most computers, an assignment like $x := x+1$ is not an atomic action. It is usually executed through copying via an internal register. For exercise I.2, the result looks like the program below.

$i := 0;$
while $i \neq 100$ do
    $r := x;$
    $r := r+1;$
    $x := r;$
    $i := i+1$
od
	$\|$	$j := 0;$
		while $j \neq 100$ do
		    $s := x;$
		    $s := s+1;$
		    $x := s;$
		    $j := j+1$
		od

If x is initially 0, what are the possible final values of x?

Exercises

I.6 The order of the statements in Peterson's algorithm is of crucial importance. Show that the program is wrong if we swap the assignments to *t* and to *bX* (and the assignments to *t* and b*Y*) by giving a counter example (a partial trace).

Is the algorithm fair?

Answers to exercises 2INC0: I.1

To some extent, as long as the number of additions of $t := Y$ is finite. It increases the number of times process Y can "overtake" process X, and therefore reduces the waiting time of Y when X is slow and Y is fast. If there are N occurrences of $t := Y$, *then Y can take at most N consecutive turns before X executes.* It does not, however, ensure minimal waiting.

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN

Answers to exercises 2INC0: I.2

$i := 0;$
while $i \neq 100$ **do**
  $x := x + 1;$
  $i := i + 1$
od

$\|$

$j := 0;$
while $j \neq 100$ **do**
  $x := x + 1;$
  $j := j + 1$
od

Shared: x

Local: i, j

Outcome: depends on atomicity of $x := x+1$

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN

Answers to exercises 2INC0: I.3

$i := 0;$ **while** $i \neq 100$ **do** $\quad x := x + 1;$ $\quad i := i + 1$ **od**	$\|\|$	$j := 0;$ **while** $j \neq 100$ **do** $\quad x := x + 1;$ $\quad j := j + 1$ **od**

Outcome: if *x := x+1* is considered to be atomic and x is initially 0 then x=200 at the end of the program execution.

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN

Answers to exercises 2INC0: I.4

Assuming atomic assignments, the following traces are possible: (P moves, Q moves**)**

a. $\{x = 0\}\ x := 1;\ x := 2\ \{x = 2\}$

$\{x = 0\}\ x := 2;\ x := 1\ \{x = 1\}$

b. $\{x = 0\}\ x := x + 1;\ x := x + 2\ \{x = 3\}$

$\{x = 0\}\ x := x + 2;\ x := x + 1\ \{x = 3\}$

c. $\{x = 0\}\ y := x;\ x := y + 1;\ x := x + 1\ \{x = 2\}\{y = 0\}$

$\{x = 0\}\ y := x;\ x := x + 1;\ x := y + 1\ \{x = 1\}\{y = 0\}$

$\{x = 0\}\ x := x + 1;\ y := x;\ x := y + 1\ \{x = 2\}\{y = 1\}$

Answers to exercises 2INC0: I.5i

$i := 0;$
while $i \neq 100$ **do**
    $r := x;$
    $r := r + 1;$
    $x := r;$
    $i := i + 1$
od

$\|$

$j := 0;$
while $j \neq 100$ **do**
    $s := x;$
    $s := s + 1;$
    $x := s;$
    $j := j + 1$
od

Post: $2 \leq x \leq 200$

Answers to exercises 2INC0: I.5ii

For $0 \leq m, n \leq 99$, consider the annotated trace

See lecture slides for a visual representation of this trace

$$
\begin{array}{l}
\quad \{x = 0\} \\
(i := 0)(j := 0) \\
\quad \{x = 0 \wedge i = 0 \wedge j = 0\} \\
((i \neq 100)(r := x)(r := r{+}1)(x := r)(i := i{+}1))^{n} \\
\quad \{x = n \wedge i = n \wedge j = 0\} \\
(j \neq 100)(s := x) \\
\quad \{x = n \wedge i = n \wedge j = 0 \wedge s = n\} \\
((i \neq 100)(r := x)(r := r{+}1)(x := r)(i := i{+}1))^{99-n} \\
\quad \{x = 99 \wedge i = 99 \wedge j = 0 \wedge s = n\} \\
(s := s{+}1)(x := s)(j := j{+}1) \\
\quad \{x = n{+}1 \wedge i = 99 \wedge j = 1\}
\end{array}
$$

Answers to exercises 2INC0: I.5iii

$$
\begin{array}{l}
\quad \{x = n{+}1 \wedge i = 99 \wedge j = 1\} \\
((j \neq 100)(s := x)(s := s{+}1)(x := s)(j := j{+}1))^{m} \\
\quad \{x = n{+}m{+}1 \wedge i = 99 \wedge j = m{+}1\} \\
(i \neq 100)(r := x) \\
\quad \{x = n{+}m{+}1 \wedge i = 99 \wedge j = m{+}1 \wedge r = n{+}m{+}1\} \\
((j \neq 100)(s := x)(s := s{+}1)(x := s)(j := j{+}1))^{99-m} \\
\quad \{x = n{+}100 \wedge i = 99 \wedge j = 100 \wedge r = n{+}m{+}1\} \\
(r := r{+}1)(x := r)(i := i{+}1) \\
\quad \{x = n{+}m{+}2 \wedge i = 100 \wedge j = 100\} \\
(i = 100)(j = 100) \\
\quad \{x = n{+}m{+}2\}
\end{array}
$$

It follows from the postcondition that any value from 2 upto and including 200 is a possible final value of x.

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN

Tanir Ozcelebi

Answers to exercises 2INC0: I.6i

We discuss the four properties required for correctness:

Deadlock can not occur because

$$
\begin{array}{ll}
 & P_X\ blocked \ \wedge\ P_Y\ blocked \\
\equiv & bY \wedge t \neq X \ \wedge\ bX \wedge t \neq Y \\
\Rightarrow & t \neq X \wedge t \neq Y \\
\equiv & false
\end{array}
$$

Minimal waiting is ensured

- Each component can take an arbitrary number of turns without the other one taking a turn.

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN

Answers to exercises 2INC0: I.6ii

Fairness (in competition) is ensured

If bX holds and process Y makes progress, then eventually a state will occur in which $t = X$ and Y will be blocked. In this state X will eventually execute its critical section

Note that fairness is about infinite traces.

Rudolf Mak, r.h.mak@tue.nl
TU/e , Computer Science, SAN

Answers to exercises 2INC0: I.6iii

Mutual exclusion may be violated

Example trace: assuming bX and bY are initially false

(Px: t:=Y) (Py: t=X) (Py: bY:=true) (Py: !bX=true) {Py enters its critical section}
(Px: bX:=true) (Px: !(t!=X)) {Px enters its critical section}